Die Pong-Platine als Digitalthermometer

Wenn man Heizkosten sparen will, braucht man ein Thermometer!
Falls man sich nur auf Erfahrungswerte verlässt, sind unweigerlich manche Räume überheizt, während man in anderen Zimmern fröstelt. Wenn man ein Gerät hat, das außerdem sehr schnell auf Temperaturänderungen reagiert, kann man zusätzlich Kältebrücken oder kühle Zugluft schnell lokalisieren.

Entsprechende Sensoren sind preisgünstig verfügbar. Im einfachsten Fall kann ein NTC eingesetzt werden, dieser müsste dann aber genau kalibriert werden um eine akzeptable Anzeigegenauigkeit erreichen zu können. Obwohl die Verarbeitung der zugehörigen nichtlinearen Temperaturfunktion für den ATmega keine Herausforderung wäre, ist es wesentlich einfacher, linearisierte Sensoren einzusetzen.

Klassische Beispiele hierfür sind der LM337 von National oder der AD22100 von Analog Devices. Die Datenblätter zu diesen Sensoren können von den Hersteller-Homepages kostenlos herunter geladen werden.

Der AD22100 besitzt ein einfaches 3-poliges Transistorgehäuse und kann sehr einfach mit der PONG-Platine verbunden werden: Positive Versorgungsspannung an P1, Masse an P4 und Signalausgang an P2 (s. a. Abb. weiter unten)

Mit zwei einfachen Konstanten (slope und offset) ergibt damit sich ein sehr brauchbares Thermometer. Dieses kann bei Bedarf durch Einzelkalibrierung sogar noch weiter präzisiert werden.

Das zugehörige C-Programm ist wie folgt aufgebaut. Nach dem Einbinden aller erforderlichen Include-Dateien werden die notwendigen Variablen deklariert. Diese beschränken sich auf den numerischen Temperaturwert selbst, ein Character-Array für die spätere ASCII-Darstellung des Temperaturwertes auf dem Display und die bereits erwähnten Kalibrationskonstanten „slope“ und „offset“.

In der main-Unit werden zunächst die Ports festgelegt. Als analoge Eingangskanäle können die ursprünglichen Potentiometereingänge (z. B. ADC6) dienen, dann hat man keine Probleme beim anschließen des Sensors.

Dann wird der Timer für den Display-Refresh initialisiert.

Nach dem Start des ADCs wird in der while(1)-Hauptschleife

- die kontinuierliche Abfrage des ADCs,

- die Umrechnung des ADC-Ergebnisses in einen Temperaturwert und
- die Anzeige des Wertes in Grad Celsius auf der LED-Matrix

ausgeführt.

```
// ThermoPONG

#define F_CPU 8000000UL

#include <stdlib.h>
#include <inttypes.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/pgmspace.h>
#include <stdio.h>
#include "pong_number_display.h"

unsigned long   t;              // temperature
char            text[3];        // display text
unsigned int    offset = 281;   // offset for AD22100
unsigned int    slope  = 217;   // slope for AD22100

int main(void)
{       // port configuration
        DDRC = 0x0f;   //  PORTC = AD-Input
        DDRB = 0xff;   //  PortB = Output
        DDRD = 0xff;   //  PortD = Output

        // initialize timer for displaying numbers
        TCCR0 |= _BV(CS01);    // F_CPU/8 as interrupt clock
        TIFR  |= _BV(TOV0);    // clear overflow flag (TOV0)
        TIMSK |= _BV(TOIE0);   // timer0 creates overflow interrupt

        // initialize ADC
        ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1);     // Prescaler=64 (8 MHz/64=125 kHz)
        ADMUX |= (1<<REFS0) | (1<<REFS1);                 // internal reference
        ADCSRA |= (1<<ADSC);                              // single conversion
        sei();

        while (1)
        {       ADMUX = 0x06;                       // select channel (ADC6)
                ADCSRA |= (1<<ADSC);                // single conversion
                while ( ADCSRA & (1<<ADSC) );       // wait for ADC ready
                t = (ADCW-offset)*slope/1000;       // scale result
                itoa(t,text,10);                    // convert result to ASCII
                if (t<10) WriteMatrix(text,6,3);    // display result at correct position
                else WriteMatrix(text,2,3);
                _delay_ms(200);
                ClearMatrix();
        }
}
```

Obwohl der Sensor prinzipiell bis –40 °C messen könnte, ist der Anzeigebereich durch die vereinfachte Zahlenkonvertierung auf positive Werte beschränkt.

Die Anzeigesteuerung erfolgt über die Include-Datei pong_number_display.h mit Ziffern im Format von 5 (Zeilen) x 3 (Spalten) = 15 Pixel.

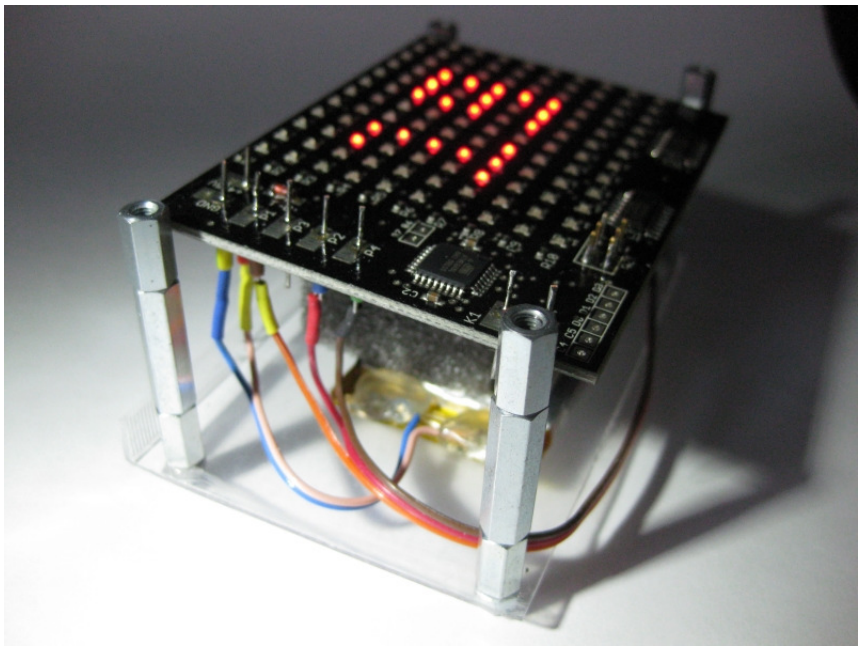
Hierzu nur eine kurze Erläuterung: die Ziffern sind als Binärdaten abgelegt, diese lauten z. B. für die „Null“: 0b11111, 0b10001, 0b11111 und geben die Belegung der drei Spalten an. Trägt man die Spalten nacheinander auf, ergibt sich das nachfolgende Muster. Wenn man dann noch die (dunklen) Nullen weg lässt und die Einsen durch Punkte ersetzt, kann man schon die Null erkennen:

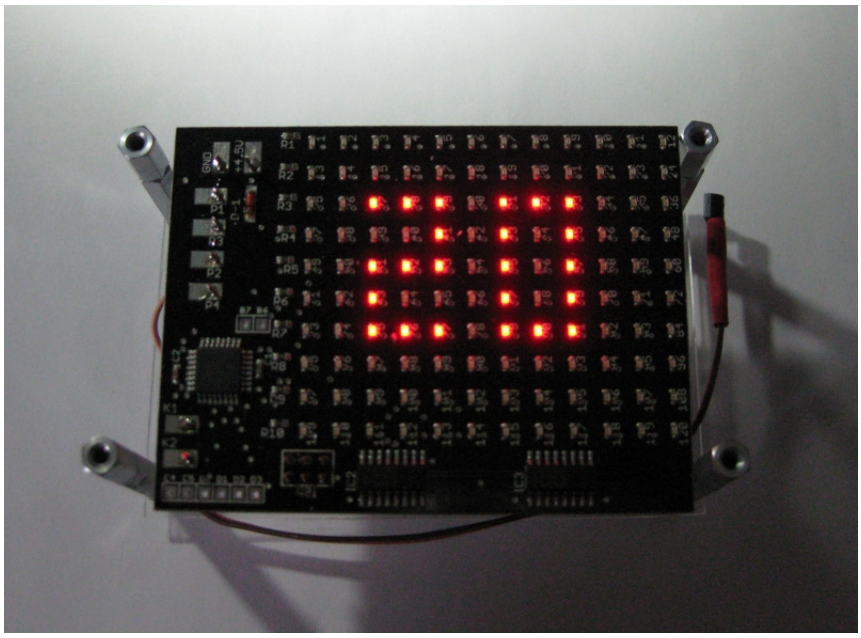
1 1 1	●●●	1 1 0	●●
1 0 1	● ●	0 1 0	●
1 0 1	● ●	0 1 0	●
1 0 1	● ●	0 1 0	●
1 1 1	●●●	1 1 1	●●●

Entsprechend lautet das Bitmuster für die „Eins“ 0b10001, 0b11111, 0b10000 usw.

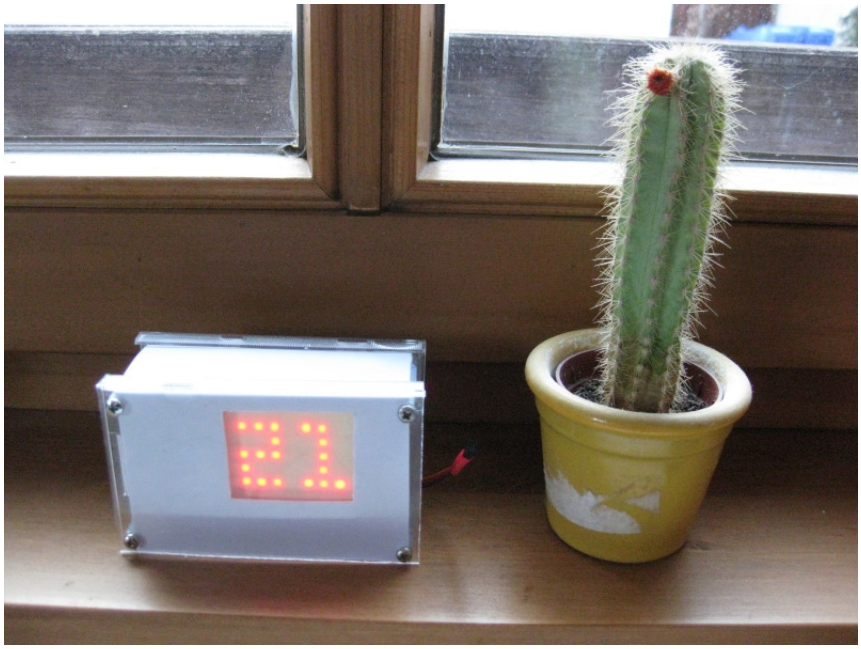
Die einzelnen Subroutinen in der pong_number_display.h-Datei dienen dem Löschen der Anzeigematrix, der Zahlenausgabe, der Initialisierung und dem Refresh.

Nach dem Einbau der Platine in ein kleines Gehäuse verfügt man über ein vollwertiges Zimmerthermometer. Die folgenden Abbildungen zeigen das Gerät in geöffnetem Zustand





und vollständig zusammengebaut im praktischem Einsatz.



Alle erforderlichen Dateien, das Hauptprogramm Thermo.c, die Include-Datei zur Display-Steuerung, pong_number_display.h sowie ein kompiliertes Hex-File Thermo.hex können als Thermo.zip **hier** herunter geladen werden.